DOCKET NO.: 98-MET-069 (STMI01-01012) **PATENT**
Customer No.: 30425

In re application of	:	David L. Isaman
U.S. Serial No.	:	09/443,160
Filed	:	November 19, 1999
For	:	SYMBOLIC STORE-LOAD BYPASS
Group No.	:	2183
Examiner	:	Idriss N. Alrobaye
Confirmation No.	:	6854

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

The Appellant has appealed to the Board of Patent Appeals and Interferences from the decision of the Examiner dated July 10, 2009, finally rejecting Claims 2-3, 6-13 and 16-21. The Appellant filed a Notice of Appeal and a Pre-Appeal Brief Request for Review on October 13, 2009. A Notice of Panel Decision from Pre-Appeal Brief Review was mailed on November 30, 2009, and set at least a one-month period for filing this Appeal Brief.

The Commissioner is hereby authorized to charge any additional fees (including any extension of time fees) connected with this communication or credit any overpayment to Deposit Account No. 50-0208.

TABLE OF CONTENTS

Real Party in Interest

The real party in interest is the assignee of this application is STMicroelectronics, Inc. as indicated by:

(1) an assignment recorded on January 24, 2000 in the Assignment Records of the United States Patent and Trademark Office at Reel 010517, Frame 0988; and

(2) a merger recorded on August 2, 2001 in the Assignment Records of the United States Patent and Trademark Office at Reel 012036, Frame 0306.

Related Appeals or Interferences

There are no known appeals or interferences that will directly affect, be directly affected by, or have a bearing on the Board's decision in this pending appeal.

Status of Claims

Claims 2-3, 6-13 and 16-21 are pending and rejected by the Office Action dated July 10, 2009 and Advisory Action of September 29, 2009. Claims 1, 4-5 and 14-15 were previously cancelled. Claims 2-3, 6-13 and 16-21 are presented for appeal. A complete and current listing of Claims 1-21 is included in Appendix A.

Status of Amendments after Final

No amendments were submitted and refused entry after the Office Action dated July 10, 2009.

Summary of Claimed Subject Matter

The following summary refers to disclosed embodiments and their advantages but does not delimit any of the claimed inventions.

In General

The present application is directed, in general, to microprocessor design.[1]

Support for Independent Claims

Note that, per 37 C.F.R. § 41.37, only the independent claims are discussed in this section, as well as any claims including means-plus-function language that are argued separately below. In the arguments below, however, various dependent claims may also be discussed and distinguished from the prior art. The discussion of the claims is for illustrative purposes and is not intended to affect the scope of the claims.

Claim 2 recites a pipelined microprocessor detecting a first instruction using first base and offset address values to load data from a first memory location that was previously stored to[2], wherein the first instruction is detected based upon the first base and offset address values and without computing a memory address equaling the first base address value added to the offset address value in detecting the first instruction.[3]

Claim 12 recites a method for operating a pipelined microprocessor, comprising:

1 See Specification, page 1, lines 1-5.
2 See Specification, page 8, line 20 – page 9, line 7; page 9, lines 13-32; page 11, line 8 – page 12, line 11.
3 See Specification, page 4, lines 8-16; page 8, line 20 – page 9, line 7; page 9, lines 13-22; page 11, line 8 – page 12, line 11.

detecting, in the pipelined microprocessor, a first instruction using first base and offset address values to load data from a first memory location that was previously stored to.[4] The first instruction is detected based upon the first base and offset address values and without computing a memory address equaling the first base address value added to the offset address value in detecting the first instruction.[5]

Claim 20 recites a method for operating a pipelined microprocessor that includes detecting a first instruction that stores data to a first memory location where the first instruction comprising syntax for computing an effective address for the first memory location.[6] The method also includes detecting a second instruction that loads data from a second memory location where the second instruction includes syntax for computing an effective address for said second memory location.[7] The method further includes determining the syntax for the first instruction and the syntax for the second instruction.[8] Further, the method includes using the syntax for the first instruction and the syntax for the second instruction to determine a relationship between the first memory location and the second memory location, without using the effective address of the first memory location or the effective address of the second memory location to determine the relationship between the first memory location and the second memory

4 See Specification, page 8, line 20 – page 9, line 7; page 9, lines 13-32; page 11, line 8 – page 12, line 11.
5 See Specification, page 4, lines 8-16; page 8, line 20 – page 9, line 7; page 9, lines 13-22; page 11, line 8 – page 12, line 11.
6 See Specification, page 7, lines 8-21; page 9, lines 1-18.
7 See Specification, page 7, lines 8-21; page 9, lines 1-18.
8 See Specification, page 7, lines 8-21; page 9, lines 1-18; page 12, lines 8-15.

location.[9] Additionally, the method includes using the relationship to determine whether to perform one of the first instruction and the second instruction.[10]

9 See Specification, page 4, lines 8-16; page 8, line 20 – page 9, line 7; page 9, lines 13-32; page 11, line 8 – page 12, line 11.
10 See Specification, page 10, lines 1-5; page 14, lines 1-4.

<u>Grounds of Rejection to be Reviewed on Appeal</u>

1. Do Claims 2-3, 6-13 and 16-19 fail to comply with the written description requirement of 35 U.S.C. § 112 First Paragraph?

2. Are Claims 2, 12 and 20 unpatentable under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 6,216,200 to Yeager, ("*Yeager*")?

3. Are Claims 2-3, 6-13 and 16-21 unpatentable under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,666,506 to Hesson, et al., ("*Hesson*")?

Arguments

Ground of Rejection #1

Claims 2-3, 6-13 and 16-21 were rejected under 35 U.S.C. § 112 First Paragraph as failing to comply with the written description requirement.

Any analysis of whether a particular claim is supported by the disclosure in an application requires a determination of whether that disclosure, when filed, contained sufficient information regarding the subject matter of the claims as to enable one skilled in the pertinent art to make and use the claimed invention. MPEP § 2164.01, p. 2100-193 (8^{th} ed., rev. 6, September 2007). The test of enablement is whether one reasonably skilled in the art could make or use the invention from the disclosures in the patent coupled with information known in the art without undue experimentation. *Id.* A patent need not teach, and preferably omits, what is well known in the art. *Id.* The Patent Office has the initial burden of establishing a reasonable basis to question the enablement provided for the claimed invention. MPEP § 2164.04 at 2100-197. The minimal requirement for a proper enablement rejection is to give reasons for the uncertainty of the enablement. *Id.*

The Examiner contends that the Specification does not explain or show the current claim language "without computing a memory address equaling the first base address value added to the offset address value in detecting the first instruction." The Examiner states that "[t]he specification does not show a memory address equaling **the first base address value added to**

the offset address value is <u>not computed</u> in detected [sic] the first instruction.[11] The Specification shows that the referenced external memory address is not computed (Specification page 4, lines 3-6) but there is not mentioning [sic] of "first base address value **added** to the offset address value."[12]

However, the Specification clearly states:

> The invention provides a method and system for operating a pipelined microprocessor more quickly, by detecting instructions without having to actually compute the referenced external memory address.[13]
>
> The instruction decode stage 120 parses the instructions 151 to determine what types of instructions 151 they are (such as instructions 151 that load data from external memory or store data to external memory). As part of parsing the instructions 151, and **<u>in addition to determine what operations</u>** the instructions 151 command the microprocessor to perform, the instruction decode stage 120 determines the syntax of any address in external memory that the instructions 151 refer to as operands.[14] (Emphasis added)

Further, the Specification, starting at page 8, line 6, teaches that an effective address is computed by an address value added to an offset value. A first instruction can be computed from a base address value added to an offset address value, and stored. Thereafter, starting at page 9, line 9, a second instruction can be detected **<u>without having to compute the effective address</u>**. Therefore, the Specification, from page 4, line 3 through page 13, line 2, teaches that an instruction can be detected <u>without computing a memory address</u>, wherein a memory address equals the first address added to the offset value.

---

11 *Advisory Action* mailed September 29, 2009, page 2
12 *Advisory Action* mailed September 29, 2009, page 2
13 See Specification page 4, lines 3-6.
14 See Specification page 6, lines 17-22.

The Examiner has failed to consider the full sequence of steps taught by the Specification. The Examiner fails to interpret the claims in light of the specification, and the § 112 is legally and factually deficient, which is clear error.

Ground of Rejection #2

Claims 2, 12 and 20 stand rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 6,216,200 to Yeager, ("*Yeager*").

A cited prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. MPEP § 2131; *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). Anticipation is only shown where each and every limitation of the claimed invention is found in a single cited prior art reference. MPEP § 2131; *In re Donohue*, 766 F.2d 531, 534, 226 U.S.P.Q. 619, 621 (Fed. Cir. 1985).

Claim 2 recites a pipelined microprocessor that includes:

> detecting a first instruction using first base and offset address values to load data from a first memory location that was previously stored to, wherein the first instruction is detected based upon the first base and offset address values and without computing a memory address equaling the first base address value added to the offset address value in detecting the first instruction.

Yeager teaches comparing virtual addresses that are, for indexed address calculations, formed by "base+index".[15] *Yeager* expressly teaches that "dependencies" may be tracked before

15 *Yeager*, col. 9, lines 21-22.

actual calculation of the virtual address based on a "presumption of the dependency" and such dependency is dynamically corrected **once the address becomes available**.[16] The dependency is a relationship between an instruction and the operands produced by a prior instruction.[17]

In the Final Office Action mailed July 10, 2009, the Examiner asserts that *Yeager* teaches each and every element as recited and arranged in independent Claim 2. The Examiner argues that *Yeager* discloses wherein the first instruction is detected based upon the first base and offset address values ("comparison of virtual addresses") at column 30, lines 43-49.[18] The Examiner also argues that *Yeager* teaches without computing a memory address equaling the first base address value added to the offset address value in detecting the first instruction ("comparison of virtual addresses" to see if there's store-to-load dependency) at column 30, lines 43-49. The Examiner then states that "[t]here is no memory address computation equaling the first base address value added to the offset address values, thus reads on the limitation." The Examiner further contends that *Yeager*, in the Abstract, teaches "wherein the dependencies are [sic] detected before virtual address calculation."[19]

In Column 30, lines 43-49, *Yeager* only teaches a load dependency operation based on a previous store wherein information to determine a block-level load dependency that is determined based on a previous store is represented by dependency bits in the store matrix (2450). Column 30, lines 28-49 states:

3. Load Dependency on Previous Stores

16 *Yeager*, Abstract.
17 See generally, *Yeager*, col. 1, lines 29-33
18 See Final Office Action, mailed July 10, 2009, pages 5-6.
19 See Final Office Action, mailed July 10, 2009, pages 5-6.

> Whenever data is stored and then loaded from the same location, the load must get the newly stored data. A memory dependency exists between a load and a previous store if:
>
> a. Both reference the same cache block (i.e., the dependency bits indicate the same cache set;
>
> the load must have a cache hit on the same way as the store);
>
> b. Both reference the same doubleword (i.e., address bits 4:3 are equal) and;
>
> c. The byte masks have at least one byte in common.
>
> Memory dependencies at the block level are detected during tag check cycles, because the cache hit signals are required to determine the selected way (which identifies the selected block).
>
> The remaining information necessary to determine a block-level load dependency on a previous store is represented by the dependency bits in store matrix 2450, which are set based exclusively on virtual addresses. These dependency bits identify store-to-load dependencies based upon common cache set and doubleword addresses (i.e., VAdr[13:3]) and byte overlap (discussed below).[20] (*Emphasis Added*)

Here *Yeager* clearly teaches that the pervious store and the subsequent store must have the same location. Accordingly, the cited portion fails to teach detecting a first instruction using first base and offset address values to load data from a first memory location that was previously stored to. Further, *Yeager* expressly teaches that the virtual address is altered by a previous store, or a default is used and reset when the previous address is ultimately calculated.[21] *Yeager* also teaches that dependencies may be tracked based on a presumption and corrected once the actual address is computed. *Yeager* further teaches comparing virtual addresses that are, for indexed address calculations, formed by "base+index".[22] As such, *Yeager* teaches tracking a value using a default value until the actual address is calculated. Therefore, a calculation still

20 *Yeager*, col. 30, lines 28-49.
21 *Yeager*, col. 9, lines 21-22.
22 *Yeager*, col. 30, lines 28-49.

must occur. Accordingly, *Yeager* clearly does not teach or suggest "detecting a first instruction using first base and offset address values to load data from a first memory location that was previously stored to, wherein the first instruction is detected based upon the first base and offset address values and <u>without using a memory address equaling to the first address value added to the offset address value</u>."

This clearly indicates that *Yeager* does not teach or suggest a "detecting a first instruction using first base and offset address values to load data from a first memory location that was previously stored to, wherein the first instruction is detected based upon the first base and offset address values and without using a memory address equaling to the first address value added to the offset address value" as recited in independent Claim 2. The Examiner's reliance on *Yeager* to teach or suggest this element of Claim 2 is clear error.

Independent Claim 12 recites elements analogous to the elements recited in independent Claim 2. Accordingly, Claim 12 is allowable for the same or similar reasons discussed above.

Claim 20 recites a method for operating a pipelined microprocessor that includes:

> determining the syntax for the first instruction and the syntax for the second instruction;
>
> using the syntax for the first instruction and the syntax for the second instruction to determine a relationship between the first memory location and the second memory location, without using the effective address of the first memory location or the effective address of the second memory location to determine the relationship between the first memory location and the second memory location; and
>
> using the relationship to determine whether to perform one of the first instruction and the second instruction.

Here, the Examiner asserts that there is no memory calculation, in *Yeager*, equaling the first base address value added to the offset address value. This is not so. As stated herein above, *Yeager* requires a calculation. *Yeager* expressly teaches that "dependencies" may be tracked before actual calculation of the virtual address based on a "presumption of the dependency" and such dependency is dynamically corrected **once the address becomes available**.[23] The dependency is a relationship between an instruction and the operands produced by a prior instruction.[24]

This clearly indicates that *Yeager* does not teach or suggest a "using the syntax for the first instruction and the syntax for the second instruction to determine a relationship between the first memory location and the second memory location, without using the effective address of the first memory location or the effective address of the second memory location to determine the relationship between the first memory location and the second memory location" as recited in independent Claim 20. The Examiner's reliance on *Yeager* to teach or suggest this element of Claim 20 is clear error.

23 *Yeager*, Abstract.
24 See generally, *Yeager*, col. 1, lines 29-33.

Ground of Rejection #2

Claims 2-3, 6-13 and 16-21 stand rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,666,506 to Hesson, et al., ("*Hesson*").

Hesson teaches an apparatus to dynamically control the out-of-order execution of load/store instructions by detecting a store violation condition and avoiding the penalty of a pipeline recovery process.[25] *Hesson* teaches using the virtual addresses – that is, the effective addresses, as opposed to the physical or "real" addresses – of memory locations to determine correspondence of two memory locations.[26]

The Examiner asserts that *Hesson* teaches that the virtual address includes a base and effective address at column 4, line 64 through column 5, line 13.[27] This is not so. The cited portion of *Hesson* states:

> The store barrier cache 11 performs a comparison of the next instruction prefetch fetch buffer virtual address against the virtual instruction address field in each of its cache entries. In general, the store barrier hit is defined as a match of the next instruction virtual address with the store barrier cache virtual address field and the condition that the store barrier bit is asserted. There may be more than one store barrier hit within the instruction fetch buffer specified by the next instruction prefetch buffer virtual address. Therefore, the store barrier cache output that is produced is the first store barrier cache hit within the store barrier cache line that is greater than or equal to the next instruction prefetch buffer address. If a store barrier cache hit results from the next instruction prefetch buffer virtual address, then the store barrier

25 *Hesson*, Abstract.
26 See generally, *Hesson*, col. 4, line 64 – col. 5, line 13.
27 *Office Action* mailed July 10, 2009, page 6.

> cache hit control output is a logic one. If no match is found, then the store barrier cache hit control output is a logic zero.[28]

Clearly, the cited portion of *Hesson* contains no disclosure that teaches or suggest that the virtual address includes a base and effective address. The Examiner also argues that *Hesson* teaches a virtual address that includes a base and offset address. The Examiner asserts that "the bits of the virtual address that are used for comparison are considered to be equivalent to the base and offset address."[29] However, no support exists (either express or inherent) in *Hesson* for the interpretation by the Examiner that the virtual address includes a base and offset address. One skilled in the art would not interpret that "bits of a virtual address" are equivalent to a base and offset address. No basis for such an interpretation, other than to contrive a basis for rejection of the claim, exists. The Examiner fails to provide a citation illustrating where *Hesson* expressly or inherently teaches that a virtual address includes a base and an offset address. The Examiner's reliance on *Hesson* to teach wherein the first instruction is detected based upon the first base and offset address values and <u>without using a memory address equaling to the first address value added to the offset address value</u> is clear error.

Independent Claim 12 recites elements analogous to the elements recited in independent Claim 2. Accordingly, Claim 12 is allowable for the same or similar reasons discussed above. Claims 3 and 6-11 depend from independent Claim 2. Claims 13 and 16-19 depend from independent Claim 12. Claims 3, 6-11, 13 and 16-19 are allowable at a minimum due to their dependence from allowable base claims.

---

28 *Hesson*, col. 4, line 64 – col. 5, line 13.
29 *Advisory Action* mailed September 29, 2009, page 4.

In regard to Claim 20, the Examiner expressly concedes that *Hesson* does not teach using the syntax for the first instruction and the syntax for the second instruction to determine a relationship between the first memory location and the second memory location.[30] The Examiner merely offers a conclusory statement that "using the effective address of the first memory location is interpreted as using the effective address to access the first memory location. However, the Examiner has not shown where *Hesson* inherently teaches such. Additionally, the Examiner has not shown wherein *Hesson* teaches <u>using the syntax</u> for both instructions to determine a <u>relationship between memory locations</u>. The Examiners interpretation of the teachings of *Hesson* to support the anticipation rejection of Claim 20 is clear error.

Further, *Hesson* teaches using the virtual addresses of memory locations to determine correspondence of two memory locations. As stated above with respect to the rejection of Claim 2, the interpretation of "the virtual address has a base and effective address" is unsupported in *Hesson*. Therefore, *Hesson* cannot reasonably be interpreted as teaching "using the syntax for the first instruction and the syntax for the second instruction to determine a relationship between the first memory location and the second memory location, without using the effective address for the first memory location or the effective address for the second memory location." The Examiner's reliance on *Hesson* to teach this feature is clear error.

Claim 21 depends from independent Claim 20. Claim 21 is allowable at a minimum due to their dependence from allowable base claims.

---

30 *Office Action* mailed July 10, 2009, page 8.

Conclusion

The Appellant respectfully submits that the cited references are improper for reasons detailed above and requests that the rejections under § 102 be withdrawn.

Requested Relief

The Board is respectfully requested to reverse the outstanding rejections and return this application to the Examiner for allowance.

Respectfully submitted,

MUNCK CARTER, LLP

Date: December 29, 2009

William A. Munck

Registration No. 39,308
ATTORNEY FOR APPELLANT

P.O. Box 802432
Dallas, Texas 75380
(972) 628-3600 (main number)
(972) 628-3616 (fax)
E-mail: *wmunck@munckcarter.com*

APPENDIX A

CLAIMS APPENDIX

1. (Canceled).

2. (Previously Presented) A pipelined microprocessor detecting a first instruction using first base and offset address values to load data from a first memory location that was previously stored to, wherein the first instruction is detected based upon the first base and offset address values and without computing a memory address equaling the first base address value added to the offset address value in detecting the first instruction.

3. (Previously Presented) A pipelined microprocessor as claimed in claim 2 wherein the pipelined microprocessor detects a second instruction using second base and offset address values to store data into a second memory location that was previously read from, wherein the second instruction is detected based upon the second base and offset address values and without computing a memory address equaling the second base address value added to the offset address values in detecting the second instruction.

4.-5. (Canceled).

6. (Previously Presented) A pipelined microprocessor as claimed in claim 2 wherein the pipelined microprocessor examines base and offset address values used to access memory locations by store instructions that store data into the memory locations, and detects load

instructions that load data from memory locations corresponding to base and offset address values identical to the base and offset address values used by the store instructions.

7. (Previously Presented) A pipelined microprocessor as claimed in claim 3 wherein the pipelined microprocessor examines base and offset address values used to access memory locations by load instructions that load data from the memory locations, and detects store instructions that store data into memory locations corresponding to base and offset address values identical to the base and offset address values used by the load instructions.

8. (Previously Presented) A pipelined microprocessor as claimed in claim 6 wherein the pipelined microprocessor detects identical offset address values and identical base address values in at least one register within the pipelined microprocessor.

9. (Previously Presented) A pipelined microprocessor as claimed in claim 7 wherein the pipelined microprocessor detects identical offset address values and identical base address values in at least one register within the pipelined microprocessor.

10. (Previously Presented) A pipelined microprocessor as claimed in claim 6 wherein the pipelined microprocessor comprises:

an instruction decode stage detecting load instructions that load data from memory locations corresponding to offset address values from an identical and base address values identical to offset address values and base address values used by prior store instructions that store data into the memory locations; and

a bypass element sending a bypass signal to an instruction execution stage of the pipelined microprocessor that indicates that a load instruction uses a base address value and an offset address value identical to a base address value and an offset address value used by a prior store instruction.

11. (Previously Presented) A pipelined microprocessor as claimed in claim 7 wherein the pipelined microprocessor comprises:

an instruction decode stage detecting store instructions that store data into memory locations using offset address values and base address values identical to offset address values and base address values used by prior load instructions that load data from memory locations; and

a bypass element sending a bypass signal to an instruction execution stage of the pipelined microprocessor that indicates that a store instruction uses a base address value and an offset address value identical to a base address value and an offset address value used by a prior load instruction.

12. (Previously Presented) A method for operating a pipelined microprocessor, comprising: detecting, in the pipelined microprocessor, a first instruction using first base and offset address values to load data from a first memory location that was previously stored to, wherein the first instruction is detected based upon the first base and offset address values and without computing a memory address equaling the first base address value added to the offset address value in detecting the first instruction.

13. (Previously Presented) A method for operating a pipelined microprocessor as claimed in claim 12, further comprising:

detecting, in the pipelined microprocessor, a second instruction using second base and offset address values to store data into a second memory location that was previously read from, wherein the second instruction is detected based upon the second base and offset address values and without computing a memory address equaling the second base address value added to the offset address values in detecting the second instruction.

14.-15. (Canceled).

16. (Previously Presented) A method for operating a pipelined microprocessor as claimed in claim 12, further comprising:

examining, in the pipelined microprocessor, base and offset address values used to access memory locations by store instructions that store data into the memory locations; and

detecting load instructions that load data from memory locations corresponding to base and offset address values identical to the base and offset address values used by the store instructions.

17. (Previously Presented) A method for operating a pipelined microprocessor as claimed in claim 13, further comprising:

examining, in the pipelined microprocessor, base and offset address values used to access memory locations by load instructions that load data from memory locations; and

detecting said instructions that store data into memory locations corresponding to base and offset address values identical to the base and offset address values used by the load instructions.

18. (Previously Presented) A method for operating a pipelined microprocessor as claimed in claim 16, further comprising:

detecting, in an instruction decode stage of the pipelined microprocessor, load instructions that load data from memory locations corresponding to offset address values and base address values identical to offset address values and base address values used by prior store instructions that store data into the memory locations; and

sending a bypass signal from a bypass element to an instruction execution stage of the pipelined microprocessor, wherein the bypass signal indicates that a load instruction uses a base address value and an offset address value identical to a base address value and an offset address value used by a prior store instruction.

19. (Previously Presented) A method for operating a pipelined microprocessor as claimed in claim 17, further comprising:

detecting, in an instruction decode stage of the pipelined microprocessor, store instructions that store data into memory locations using offset address values and base address values identical to offset address values and base address values used by prior load instructions that load data from memory locations; and

sending a bypass signal from a bypass element to an instruction execution stage of the pipelined microprocessor, wherein the bypass signal indicates that a load instruction uses a base address value and an offset address value identical to a base address value and an offset address value used by a prior store instruction.

20. (Previously Presented) A method for operating a pipelined microprocessor, comprising:

detecting a first instruction that stores data to a first memory location, the first instruction comprising syntax for computing an effective address for the first memory location;

detecting a second instruction that loads data from a second memory location, the second instruction comprising syntax for computing an effective address for said second memory location;

determining the syntax for the first instruction and the syntax for the second instruction;

using the syntax for the first instruction and the syntax for the second instruction to determine a relationship between the first memory location and the second memory location, without using the effective address of the first memory location or the effective address of the second memory location to determine the relationship between the first memory location and the second memory location; and

using the relationship to determine whether to perform one of the first instruction and the second instruction.

21. (Previously Presented) A method for operating a pipelined microprocessor as claimed in claim 20, wherein the syntax for the first instruction and the syntax for the second instruction refer to an identical memory location.

APPENDIX B -

Evidence Appendix

Not Applicable -- To the best knowledge and belief of the undersigned attorney, there are none.

APPENDIX C -

Related Proceedings Appendix

Not Applicable -- To the best knowledge and belief of the undersigned attorney, there are none.

US006216200B1

(12) **United States Patent**
Yeager

(10) **Patent No.:** **US 6,216,200 B1**
(45) **Date of Patent:** ***Apr. 10, 2001**

(54) **ADDRESS QUEUE**

(75) Inventor: **Kenneth Yeager**, Sunnyvale, CA (US)

(73) Assignee: **MIPS Technologies, Inc.**, Mountain View, CA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/404,625**

(22) Filed: **Mar. 14, 1995**

Related U.S. Application Data

(63) Continuation-in-part of application No. 08/324,129, filed on Oct. 14, 1994, now abandoned.

(51) **Int. Cl.**[7] **G06F 9/20**

(52) **U.S. Cl.** **711/100**; 711/3; 711/202; 711/104; 711/118

(58) **Field of Search** 395/800, 375, 395/24, 491, 650, 427, 440, 413, 412, 104, 403, 431, 392, 676, 389; 711/3, 5, 4, 202, 203, 150, 100

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,466,061	*	8/1984	DeSantis et al.	395/650
5,134,561	*	7/1992	Liptay	395/491
5,201,057	*	4/1993	Uht	395/800
5,377,336	*	12/1994	Eicknemeyer et al.	395/375
5,467,473	*	11/1995	Kahle et al.	395/800
5,471,593	*	11/1995	Branigin	395/24
5,488,729	*	1/1996	Vegesna et al.	395/800
5,511,175	*	4/1996	Favor et al.	395/375

OTHER PUBLICATIONS

Cocke, et al., "The Evolution of RISC Technology at IBM," IBM J. Res. Develop., vol. 34, No. 1 p. 4–36 (Jan. 1990).

Grohoski, "Machine Organization of the IBM RISC System/6000 Processor," IBM J. Res. Develop., vol. 34, No. 1 p. 37–58 (Jan., 1990).

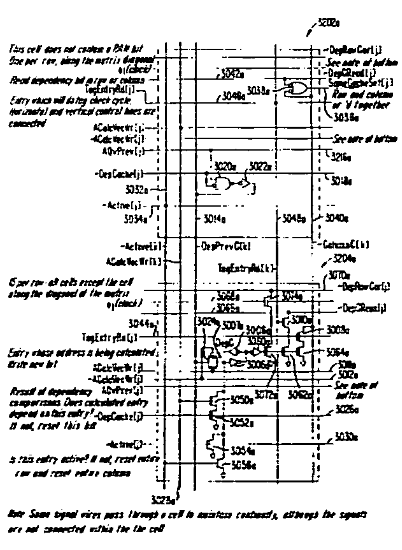
* cited by examiner

Primary Examiner—David K. Moore
Assistant Examiner—Than V. Nguyen

(57) **ABSTRACT**

An address queue in a processor having the capability to track memory-dependencies of memory-access instructions is disclosed. The queue includes a first matrix of RAM cells that tracks a first dependency relationship between a plurality of instructions based upon matching virtual addresses (that identify a common cache set) and the order of instructions in the queue. To facilitate out-of-order instruction execution, dependencies may be tracked before virtual addresses are actually calculated based upon a presumption of dependency. Such dependency is dynamically corrected as addresses become available. The same comparison mechanism used to determine matching virtual addresses for the dependency relationship may also be used to read status bits of a cache set being accessed. The queue also includes a second matrix of RAM cells that tracks a second dependency relationship between a plurality of instructions based upon matching virtual addresses (that identify a common cache set, common doubleword and overlapping byte), the order of instructions in the queue and instruction type. Also disclosed is a method for processing memory instructions that uses a single comparison step between first and second virtual addresses (calculated from instructions) to indicate a dependency relationship between the instructions and to read memory status bits. The status bits are read to determine accessibility of a way within an addressed cache set.

22 Claims, 46 Drawing Sheets



US005666506A

United States Patent [19]

Hesson et al.

[11] Patent Number: 5,666,506

[45] Date of Patent: Sep. 9, 1997

[54] APPARATUS TO DYNAMICALLY CONTROL THE OUT-OF-ORDER EXECUTION OF LOAD/STORE INSTRUCTIONS IN A PROCESSOR CAPABLE OF DISPATCHNG, ISSUING AND EXECUTING MULTIPLE INSTRUCTIONS IN A SINGLE PROCESSOR CYCLE

[75] Inventors: **James Henry Hesson**; **Jay LeBlanc**; **Stephen J. Ciavaglia**, all of Chittenden County, Vt.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **440,025**

[22] Filed: **May 12, 1995**

Related U.S. Application Data

[62] Division of Ser. No. 328,185, Oct. 24, 1994, abandoned.

[51] **Int. Cl.**^{6} **G06F 9/38**

[52] **U.S. Cl.** **395/392**; 395/394

[58] **Field of Search** 395/375, 800, 395/585, 394, 392

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,481,576	11/1984	Bicknell	395/421.07
5,261,071	11/1993	Lyon	395/467
5,323,489	6/1994	Bird	395/494
5,325,505	6/1994	Hoffecker et al.	395/612
5,442,757	8/1995	McFarland et al.	395/394
5,511,175	4/1996	Favor et al.	395/392
5,530,958	6/1996	Agarwal et al.	395/403

Primary Examiner—William M. Treat
Assistant Examiner—Kenneth R. Coulter
Attorney, Agent, or Firm—Whitham, Curtis, Whitham & McGinn; Susan M. Murray

[57] **ABSTRACT**

An apparatus to dynamically controls the out-of-order execution of load/store instructions by detecting a store violation condition and avoiding the penalty of a pipeline recovery process. The apparatus permits a load and store instruction to issue and execute out of order and incorporates a unique store barrier cache which is used to dynamically predict whether or not a store violation condition is likely to occur and, if so, to restrict the issue of instructions to the load/store unit until the store instruction has been executed and it is once again safe to proceed with out-of-order execution. The method implemented by the apparatus delivers performance within one percent of theoretically possible with apriori knowledge of load and store addresses.

1 Claim, 4 Drawing Sheets